UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dellInformazione ed Elettrica e Matematica applicata

Corso di Laurea Magistrale in Ingegneria Informatica

Corso I-Z : Gruppo 23

PW: Algoritmi e Protocolli per la Sicurezza

Docenti
Prof. Carlo Mazzocca
Prof. Francesco Cauteruccio

Autori
Gabriele Imparato
Michele Penna

Anno Accademico 2024/2025

Indice

Capitolo 1

WP 1: Modello

Responsabile WP1:	Gabriele Imparato

Il presente modello operativo descrive il contesto e le dinamiche di scambio di credenziali accademiche digitali tra due università coinvolte in un programma Erasmus. L'obiettivo è garantire che lo studente possa trasferire in modo sicuro e verificabile le informazioni relative alla propria carriera, utilizzando strumenti digitali affidabili come *wallet*, *firme digitali* e *blockchain*.

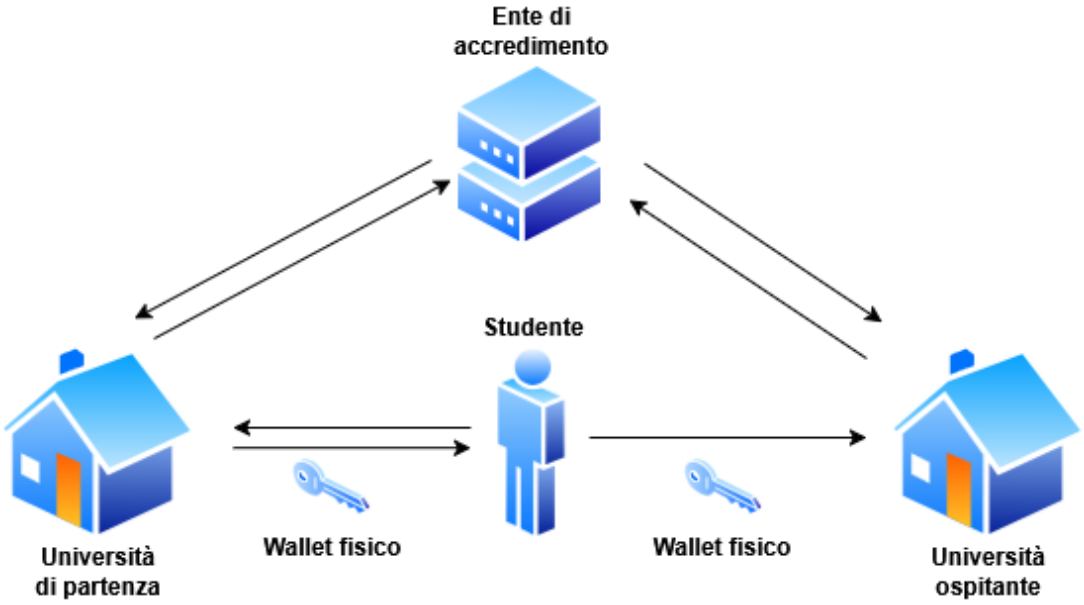

Figura 1.1: Condivisione selettiva di credenziali accademiche tra Université de Rennes e Università di Salerno.

1.1 Attori

Il modello operativo analizzato utilizza una consegna di credenziali per uno studente erasmus dall'università di provenienza di quest'ultimo a quella di destina-

zione, quindi gli attori riscontrati nella specifica da realizzare sono stati divisi in onesti e disonesti (*Threat model*).

1.1.1 Attori onesti

- **Studente Erasmus**: rappresenta il soggetto centrale del processo di mobilità internazionale. Ha la necessità di ottenere credenziali accademiche digitali valide, conservarle in modo sicuro e presentarle facilmente alle istituzioni coinvolte. Deve inoltre poter controllare quali dati condividere e con chi, garantendo la propria privacy e la sicurezza delle informazioni personali;

- **Università di partenza** *Université de Rennes*: è l'istituzione accademica di origine dello studente. È responsabile dell'emissione delle credenziali digitali che attestano il superamento degli esami e delle attività formative. Tali credenziali devono rispettare standard interoperabili, essere firmate digitalmente e riconosciute a livello internazionale;

- **Università ospitante** *Università di Salerno*: accoglie lo studente durante il suo periodo di studio all'estero. Deve poter ricevere e verificare l'autenticità delle credenziali presentate, assicurandosi che provengano da un'autorità accademica riconosciuta, al fine di procedere con il riconoscimento degli esami e dei crediti formativi;

- **Sistema di wallet fisico**: è lo strumento tecnico utilizzato dallo studente per gestire le proprie credenziali accademiche. Deve garantire una conservazione sicura (ad esempio tramite crittografia), permettere la selezione dei dati da condividere (secondo il principio di minimizzazione) e assicurare che la presentazione delle credenziali sia controllata, verificabile e conforme alle normative vigenti in materia di protezione dei dati;

- **Enti di certificazione**: sono autorità fidate che validano le identità digitali degli attori coinvolti (università, studenti, provider del wallet) e garantiscono l'autenticità delle credenziali. Rivestono un ruolo fondamentale nella costruzione di un ecosistema basato sulla fiducia, trasparente e resistente a possibili frodi.

1.1.2 Threat Model

Il modello di minaccia prevede diversi scenari realistici. Un primo rischio riguarda lo studente disonesto, che potrebbe tentare di falsificare la propria carriera accademica dichiarando esami mai sostenuti. A livello di comunicazione tra atenei, è possibile immaginare la presenza di un attaccante esterno che intercetti i messaggi per rubare identità o alterare informazioni: dal semplice hacker fino a servizi segreti internazionali.

L'università stessa potrebbe rilasciare credenziali errate o incomplete, dove ne risentirebbe la reputazione dello studente con l'ateneo di arrivo. Infine, non si

dovrebbe escludere attività maliziose da parte di studenti non autorizzati che potrebbero introdurre vulnerabilità nel sistema.

Quindi, possiamo riassumere gli attori malevoli, con i loro obiettivi e risorse come:

1. **Studente malintenzionato**: un soggetto che cerca di ottenere vantaggi indebiti nel sistema accademico;
    - *Obiettivi*: creare, modificare o presentare credenziali accademiche false al fine di ottenere il riconoscimento di titoli non meritati;
    - *Risorse*: pieno controllo sul proprio portafoglio digitale e possibilità di scegliere quando e come presentare le credenziali;
2. **Università malintenzionata o negligente**: ente accademico che agisce in modo fraudolento oppure che non adotta adeguate misure di sicurezza;
    - *Obiettivi*: emettere credenziali false o manipolate per interessi propri, oppure raccogliere dati personali degli studenti senza un consenso informato;
    - *Risorse*: accesso completo all'infrastruttura informatica di ateneo, inclusi i sistemi di gestione delle credenziali e le informazioni sugli studenti;
3. **Enti governativi malintenzionati**: autorità statali che abusano del proprio potere per fini non trasparenti o repressivi;
    - *Obiettivi*: monitorare le attività degli studenti, raccogliere dati sensibili o ostacolare la libera circolazione delle informazioni;
    - *Risorse*: accesso privilegiato alle infrastrutture di rete, potere legislativo e tecnologie avanzate di sorveglianza;
4. **Attaccanti esterni (terze parti)**: hacker o organizzazioni con l'obiettivo di compromettere l'integrità e la riservatezza del sistema;
    - *Obiettivi*: intercettare comunicazioni, accedere a credenziali o informazioni personali, oppure manipolare dati durante la trasmissione;
    - *Risorse*: tecniche di phishing, malware, exploit di vulnerabilità nei sistemi di comunicazione o autenticazione.

1.2 Proprietà del sistema

Dato il modello operativo e le minacce identificate, è fondamentale definire le proprietà di sicurezza che il sistema deve garantire per essere considerato robusto e affidabile. Queste proprietà rappresentano i requisiti fondamentali che guideranno la progettazione della soluzione nel WP2. Esse assicurano che il sistema si comporti come previsto anche in presenza degli avversari descritti nel Threat Model.

- **Integrità**:
  - **Definizione**: il destinatario deve essere in grado di capire se il messaggio ricevuto non sia stato alterato;
  - **Contesto Erasmus**: l'integrità è garanzia che i dati accademici non possano essere manipolati né durante la trasmissione né mentre sono archiviati nel wallet dello studente. Per l'Università ospitante, è cruciale avere la certezza che un voto non venga fraudolentemente modificato da uno studente malintenzionato. Allo stesso modo, un attaccante esterno non deve poter intercettare e alterare i dati di una credenziale per danneggiare lo studente o l'istituzione. Senza questa proprietà, l'intero sistema di riconoscimento dei crediti perderebbe ogni valore;
- **Privacy**:
  - **Definizione**: la comunicazione tra due attori deve essere privata;
  - **Contesto Erasmus**: la privacy è essenziale per proteggere i dati sensibili dello studente e per rispettare normative come il GDPR. Lo studente deve poter condividere solo le informazioni strettamente necessarie (es. il superamento di un esame specifico) senza dover rivelare l'intera carriera, dati personali non pertinenti (es. income) o informazioni su borse di studio (scholarship). Questa proprietà protegge lo studente da università negligenti che potrebbero raccogliere troppi dati, da enti governativi che mirano alla sorveglianza e da attaccanti esterni interessati al furto di dati personali;
- **Autenticità**:
  - **Definizione**: il destinatario deve poter verificare che il messaggio provenga dal mittente desiderato;
  - **Contesto Erasmus**: l'autenticità assicura che una credenziale sia state emessa da un'istituzione legittima. L'Università di Salerno deve poter verificare senza ombra di dubbio che una credenziale provenga effettivamente dall'Université de Rennes e non sia stata creata da zero da uno studente malintenzionato. Questa proprietà previene la falsificazione di intere carriere accademiche e garantisce che solo le università accreditate possano rilasciare titoli validi all'interno del sistema;
- **Revocabilità**:
  - **Definizione**: deve essere possibile la rottura del legame tra la chiave pubblica e il soggetto di interesse;
  - **Contesto Erasmus**: la vita di una credenziale non è necessariamente infinita. Potrebbe essere necessario invalidarla per vari motivi: un errore amministrativo, la scoperta di un plagio, o la compromissione del wallet fisico dello studente. Il sistema deve prevedere un meccanismo

sicuro e trasparente che permetta all'università emittente di revocare una credenziale, assicurando che questa non possa più essere utilizzata fraudolentemente in futuro;

- **Non ripudio**:
  - **Definizione**: il mittente non deve essere in grado di negare di aver inviato il messaggio. Il destinatario deve poter convincere un attore terzo che il messaggio ricevuto è stato inviato dal mittente presunto (utilizzo di firme digitali);
  - **Contesto Erasmus**: il non ripudio protegge sia lo studente che le istituzioni. Se l'Università di Rennes emette una credenziale, non deve poter in seguito negare di averlo fatto, magari per coprire un errore amministrativo o per danneggiare lo studente. Questa proprietà, garantita tipicamente dalle firme digitali, crea una prova crittografica innegabile dell'emissione, che può essere verificata da chiunque.

In riferimento ai threat model, le proprietà di sicurezza possono essere messere a dura prova:

- Lo studente malintenzionato può falsificare credenziali, quindi attacca integrità, autenticità e non ripudio;
- L'università malintenzionata ha potere su tutto il sistema informatico, quindi può compromettere quasi tutte le proprietà;
- Gli enti governativi malintenzionati agiscono molto sulla privacy;
- Gli attaccanti esterni puntano su privacy, integrità e autenticità.

Riassunto nella tabella 1.1.

Attore	**Integrità**	**Privacy**	**Autenticità**	**Revocabilità**	**Non ripudio**
Studente mal.	Sì	Forse	Sì	No	Sì
Università mal.	Sì	Sì	Sì	Sì	Forse
Enti gov. mal.	Forse	Sì	Forse	No	No
Attaccanti esterni	Sì	Sì	Sì	No	Forse

Tabella 1.1: Proprietà messe a rischio dai vari threat model.

1.3 Parametri del wallet fisico

La definizione della struttura dati della credenziale è un passo fondamentale del modello, poiché stabilisce quali informazioni possono essere gestite, archiviate nel wallet e presentate selettivamente. Una struttura ben definita deve essere

abbastanza completa da coprire le necessità informative delle università, ma anche sufficientemente granulare da consentire allo studente di esercitare un controllo preciso sulla propria privacy.

Per questo motivo, i parametri della credenziale sono stati suddivisi in due categorie logiche: quelli strettamente legati al percorso accademico presso l'ateneo e quelli relativi all'identificazione personale dello studente.

1.3.1 Parametri legati all'ateneo

Questo primo gruppo di attributi definisce il profilo accademico e amministrativo dello studente all'interno dell'università di origine. Sono le informazioni che costituiscono il nucleo della carriera accademica.

```
- student_ID
- first_name
- last_name
- origin_university
- degree
- department
- course_enrollment_year
- exam_transcripts
- email
- cafeteria
- scholarship
```

Listing 1.1: Esempio di credenziale per lo studente

I parametri indicano:

- `student_ID`: identificativo univoco dello studente nell'ateneo di provenienza;
- `first_name`: nome dello studente;
- `last_name`: cognome dello studente;
- `origin_university`: nome dell'università di provenienza;
- `degree`: nome del corso di laurea frequentato dallo studente;
- `department`: nome del dipartimento del corso di laurea frequentato dallo studente;
- `course_enrollment_year`: anno di iscrizione dello studente;
- `exam_transcripts`: libretto degli esami dello studente;
- `email`: indirizzo di posta elettronica d'ateneo dello studente;
- `cafeteria`: parametro vero/falso riferito all'utilizzo della mensa universitaria dello studente;
- `scholarship`: parametro vero/falso riferito allo studente come beneficiario di borsa di studio.

1.3.2 Parametri di identificazione

Questo secondo gruppo di attributi contiene dati anagrafici e personali. Sebbene siano spesso necessari per le procedure di iscrizione formale, sono anche più sensibili e la loro condivisione deve essere gestita con ancora maggiore cautela.

```
- sex
- birthplace
- birthday
- residence
- country
- citizenship
- spoken_languages
- social_security_number
- income
```

Listing 1.2: Esempio di credenziale per lo studente

I parametri indicano:

- `sex`: sesso dello studente;
- `birthplace`: luogo di nascita dello studente;
- `birthday`: data di nascita dello studente (formato *DD/MM/AA*);
- `residence`: luogo di residenza dello studente;
- `country`: nazione di residenza dello studente;
- `citizenship`: cittadinanza dello studente;
- `spoken_languages`: lingue parlate dallo studente;
- `social_security_number`: codice fiscale (o affine) dello studente;
- `income`: reddito annuo dello studente.

Capitolo 2

WP 2: Soluzione

Responsabile WP2:	Michele Penna

2.1 Architettura del Sistema

2.1.1 Attori Coinvolti

- **Studente:** Colui che presenta, richiede e gestisce le proprie credenziali.
- **Università Emittente:** L'ente che crea, firma e rilascia le credenziali accademiche.
- **Università Verificatrice:** L'ente che riceve e verifica le credenziali presentate dallo studente.
- **Wallet dello Studente:** Un componente (ad esempio un wallet fisico) gestito dallo studente per conservare le credenziali e le relative chiavi private, e per generare le presentazioni selettive.
- **Distributed Ledger Technology (DLT):** Un registro distribuito e immutabile in cui i dati digitali vengono replicati, condivisi e sincronizzati tra più sedi, garantendo che tutti i partecipanti mantengano una copia locale dei dati (come le radici dei Merkle Tree delle credenziali e il loro stato di revoca) in modo decentralizzato e trasparente.

2.1.2 Componenti Tecnologici Chiave e Motivazioni

Per rispondere alle esigenze di decentralizzazione, privacy e controllo da parte dell'utente, abbiamo progettato un'architettura che non si affida a un'autorità centrale per la gestione e la verifica delle identità, ma utilizza i concetti di crittografia asimmetrica, firme digitali e funzioni hash per creare un sistema di fiducia distribuito. Di seguito, mostriamo nel dettaglio l'architettura scelta per affrontare il problema.

Merkle Tree e funzioni Hash

Per garantire l'integrità della credenziale e, al tempo stesso permettere una condivisione selettiva dei dati, la nostra soluzione fa utilizzo dei Merkle Tree.

Ogni credenziale accademica viene scomposta nei vari attributi atomici (es. nome, cognome, esame, voto, data, ...). Ciascun attributo viene trattato come una foglia di un albero binario. Ogni nodo padre nell'albero viene calcolato applicando una funzione hash crittografica alla concatenazione dei suoi nodi figli. Questo processo viene ripetuto ricorsivamente fino a ottenere un unico hash finale, che corrisponde alla radice (root) del Merkle Tree. Questa radice agisce come un'impronta digitale compatta e univoca dell'intera credenziale.

La sicurezza dei Merkle Tree dipende interamente dalle proprietà della funzione hash sottostante. Abbiamo pensato di utilizzare l'**algoritmo SHA-256**, caratterizzato da robustezza alle collisioni. Questa proprietà è fondamentale, perché ci assicura che sia computazionalmente difficile trovare due insiemi di attributi che originano la stessa radice nel Merkle Tree, prevenendo così la falsificazione dei dati.

Questo approccio ci consente di ottenere condivisione selettiva. Lo studente può così mostrare a un verificatore solo un sottoinsieme di attributi (*Proof of membership*). Per dimostrare che un blocco di dati è incluso nell'albero, è sufficiente mostrare i blocchi nel percorso che va da quel blocco di dati fino alla radice. Il verificatore può in questo modo verificare l'autenticità degli attributi specifici senza venire a conoscenza del resto della credenziale.

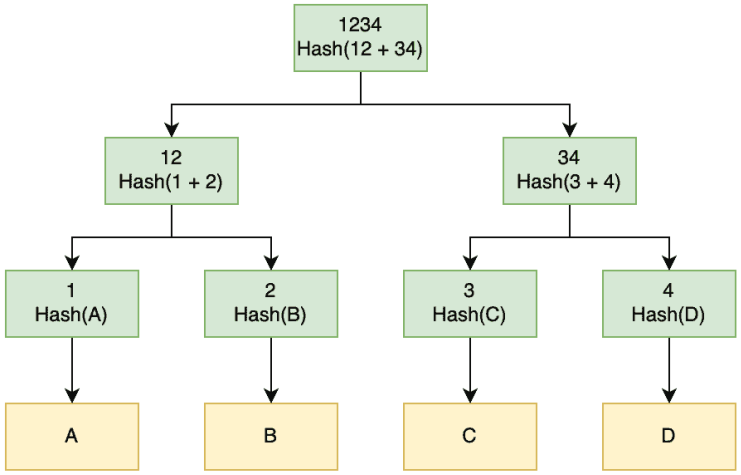

Figura 2.1: Hashing dei singoli blocchi per merkle tree.

Crittografia a Chiave Pubblica e Firme Digitali

Per legare in modo inequivocabile una credenziale alla sua università emittente e per proteggerla da modifiche, pertanto garantendone l'autenticità, utilizziamo meccanismi di firma digitale basati sulla crittografia asimmetrica. L'Università Emittente firma digitalmente la radice del Merkle Tree della credenziale utilizzando la propria chiave privata. La firma risultante, insieme alla radice dell'albero,

diventa parte integrante della credenziale che viene consegnata allo studente. In questo modo, chiunque sia in possesso della chiave pubblica dell'università può verificare la validità della firma, confermando in questo modo il principio di non ripudio.

Per implementare la nostra soluzione utilizziamo l'algoritmo di firma **RSA**, utilizzando uno schema con **padding** sicuro come **PSS**. Con la scelta proposta, andiamo a rispettare le seguenti proprietà:

- **Autenticità:** Solo l'entità in possesso della chiave privata dell'università può generare una firma valida, garantendo al verificatore la provenienza della credenziale;
- **Integrità:** La firma è calcolata sulla radice del Merkle Tree. Qualsiasi alterazione, a uno degli attributi della credenziale cambierebbe la radice, rendendo la firma invalida;
- **Non Ripudio:** L'università emittente non può negare di aver emesso la credenziale, poiché solo lei avrebbe potuto creare quella specifica firma con la sua chiave privata.

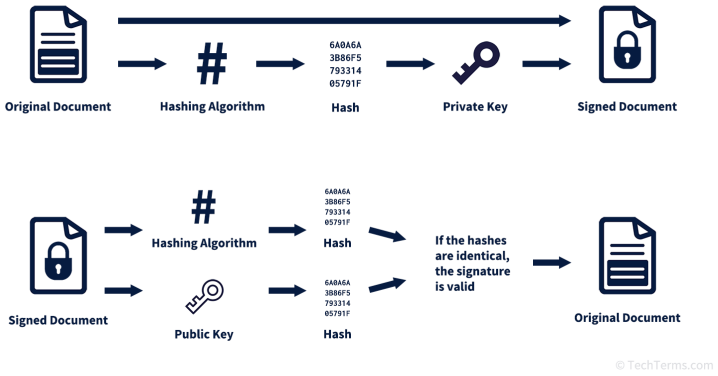

Figura 2.2: Processo tipico di firma digitale su documento.

Distributed Ledger Technology (DLT)

Per eliminare la necessità di un'autorità centrale e per creare un punto di riferimento pubblico e immutabile, la nostra architettura si appoggia a una **Distributed Ledger Technology**. La DLT è un registro distribuito, un sistema in cui i dati digitali vengono replicati, condivisi e sincronizzati tra più sedi, garantendo che tutti i partecipanti mantengano una copia locale dei dati. In linea con il principio di minimizzazione dei dati, è fondamentale sottolineare che nessun dato personale dello studente viene memorizzato sulla DLT. Il suo ruolo è limitato a registrare in modo permanente e verificabile esclusivamente le informazioni necessarie a stabilire la fiducia nel sistema:

- Le chiavi pubbliche delle università partecipanti;
- Le *Merkle Root* delle credenziali emesse, firmate dalle rispettive università;

- Le informazioni relative alla revoca di una credenziale;

Questo approccio sfrutta le proprietà intrinseche delle DLT: *decentralizzazione*, *immutabilità* e *trasparenza*.

In una **Public Key Infrastructure** (**PKI**) tradizionale la fiducia è riposta in una o più **Certification Authority** (**CA**) centralizzate. La nostra architettura, invece, utilizza la DLT per creare un sistema di fiducia decentralizzato. Invece di doversi fidare di un singolo ente, ovvero la CA, la fiducia viene distribuita e garantita dal protocollo di consenso della DLT stessa.

Il processo di verifica finale (sezione 2.2.3) si basa proprio su questo principio. Include un controllo sulla DLT per due scopi:

1. Confermare che la radice del Merkle Tree presentata sia stata effettivamente pubblicata dall'università.

2. Assicurarsi che la credenziale non risulti in una lista di revoca.

In questo modo, la fiducia viene stabilita senza ricorrere a un'autorità centrale, ma basandosi sulle garanzie fornite dal protocollo distribuito.

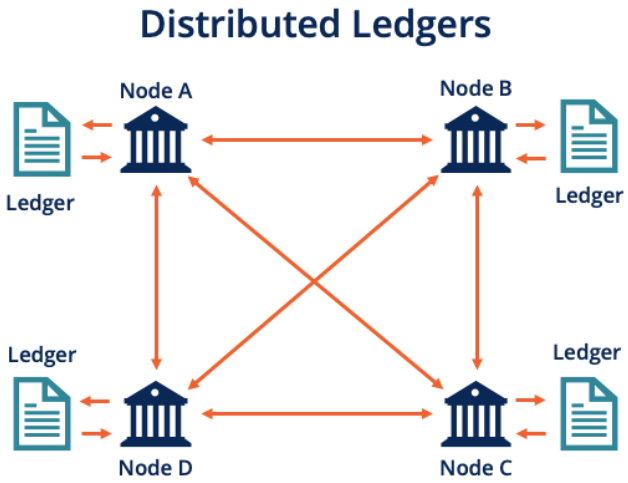

Figura 2.3: Architettura tipica di un Distribuite Ledger decentralizzato (registro distribuito).

2.1.3 Panoramica del Flusso Operativo

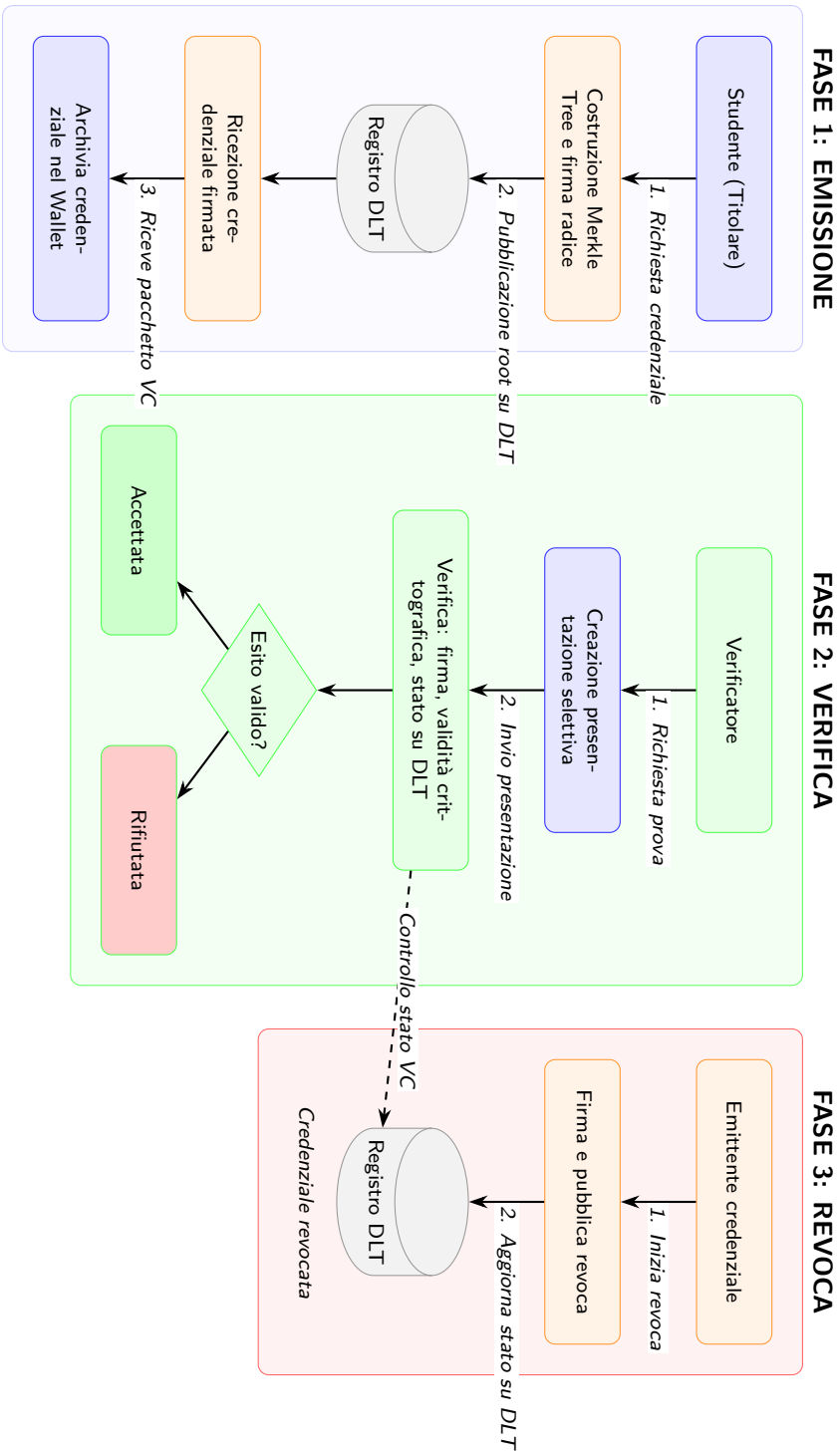

Figura 2.4: Schema del processo di emissione, verifica e revoca di una credenziale digitale basata su DLT.

Il diagramma in figura 2.4 rappresenta il **processo completo di gestione delle credenziali digitali** in tre fasi principali:

1. **Emissione** (**FASE 1**):
   (a) Lo studente (titolare della credenziale) richiede l'emissione;
   (b) Viene costruito un Merkle Tree e firmata la radice, che viene pubblicata sul DLT;
   (c) La credenziale firmata viene poi ricevuta e archiviata nel wallet digitale dello studente;
2. **Verifica o Presentazione** (**FASE 2**):
   (a) Un verificatore richiede una prova di possesso della credenziale;
   (b) Lo studente crea una presentazione selettiva, cioè mostra solo le parti necessarie della credenziale;
   (c) Il verificatore controlla la firma, la validità crittografica e verifica lo stato della credenziale sul DLT;
   (d) In base al controllo, la presentazione può essere **accettata** o **rifiutata**;
3. **Revoca** (**FASE 3**):
   - L'emittente della credenziale può iniziare la procedura di revoca, firmando e pubblicando lo stato di revoca sul DLT;
   - A seguito di questa operazione, la credenziale risulta marcata come revocata e quindi.

2.2 Protocolli di sicurezza adoperati

Definiamo i protocolli che gestiscono le credenziali. Per ognuno di essi, sono specificati attori, le precondizioni, i passi sequenziali con le relative operazioni crittografiche e le post-condizioni risultanti.

2.2.1 Protocollo di Emissione delle Credenziali

Il protocollo di emissione consente di ottenere sicurezza e affidabilità delle credenziali. È progettato per garantire che solo gli studenti legittimi ricevano credenziali corrette e crittograficamente sicure.

- **Attori Coinvolti:**
  - Studente: il richiedente della credenziale;
  - Università Emittente: l'autorità che emette e firma la credenziale;
  - Distributed Ledger Technology (DLT): il registro distribuito per la pubblicazione delle prove di emissione;

- **Pre-condizioni:**
  - Lo studente è regolarmente iscritto e ha diritto a ricevere la credenziale;
  - Lo studente possiede credenziali valide (es. username/password) per autenticarsi ai servizi online dell'Università Emittente;
  - L'Università Emittente possiede una coppia di chiavi crittografiche istituzionale (es. `PK_UE`, `SK_UE`) basata su RSA, utilizzando uno schema di firma con padding PSS. La chiave pubblica `PK_UE` è nota e pubblicamente verificabile;
- **Step:**
  1. **Autenticazione e Richiesta:** lo Studente si autentica al portale dell'Università Emittente e richiede l'emissione di una specifica credenziale digitale;
  2. **Autorizzazione e Preparazione Dati:** l'Università Emittente valida il token, verifica il diritto dello studente alla credenziale e struttura gli attributi (es. attraverso un formato JSON);
  3. **Costruzione del Merkle Tree:** l'Università Emittente costruisce un Merkle Tree dagli attributi della credenziale, calcolando la radice del Merkle Tree tramite l'algoritmo SHA-256;
  4. **Firma Digitale della Radice:** l'Università Emittente firma digitalmente la radice del Merkle Tree con la sua chiave privata `SK_UE`, garantendo autenticità, integrità e non ripudio;
  5. **Registrazione su DLT:** L'Università Emittente pubblica la radice del Merkle Tree sul DLT, creando una prova di emissione immutabile;
  6. **Consegna della Credenziale:** l'Università Emittente invia allo studente un pacchetto contenente gli attributi, la *MTR* (*Merkle Tree Root*), la *firma* e le *proof of membership*;
  7. **Verifica e Archiviazione nel Wallet:** il wallet dello studente verifica l'integrità e l'autenticità del pacchetto ricevuto prima di archiviarlo in modo sicuro;
- **Post-condizioni:**
  - Lo studente possiede una credenziale digitale sicura e auto-verificante;
  - Esiste una prova pubblica dell'emissione sulla DLT;
  - Lo studente ha il pieno controllo sulla sua credenziale;

2.2.2 Protocollo di Presentazione Selettiva (Selective Disclosure)

Questo protocollo è il cuore della funzionalità di privacy del sistema e applica il principio del minimo privilegio (*least privilege*), ovvero ogni soggetto dovrebbe

operare utilizzando un insieme minimo di privilegi (diritti di accesso) necessari per svolgere il proprio compito.

- **Attori Coinvolti:** Studente, Università Verificatrice;
- **Pre-condizioni:** lo studente possiede un `CredentialPackage` valido; l'Università Verificatrice ha definito quali attributi sono necessari; la `PK_UE` è accessibile all'Università Verificatrice;
- **Step:**
  1. **Richiesta di Presentazione:** l'Università Verificatrice richiede allo studente una prova per un insieme specifico di attributi;
  2. **Consenso dell'utente e generazione della proof of membership:** il wallet dello studente chiede il consenso esplicito e genera la proof of membership, ovvero il set di hash che permette di verificare gli attributi richiesti senza rivelare gli altri;
  3. **Creazione della Presentazione Verificabile:** il wallet assembla un pacchetto di presentazione. Invece di inviare l'intera credenziale, questo pacchetto contiene solo: gli attributi selezionati richiesti dall'Università Verificatrice, la prova crittografica (*proof of membership*) associata a quegli attributi, la radice del Merkle Tree e la firma digitale originali dell'Università Emittente, che servono a garantire l'autenticità del tutto;
  4. **Invio della Presentazione:** il wallet invia il pacchetto all'Università Verificatrice su un canale sicuro;
- **Post-condizioni:**
  - L'Università Verificatrice ha ricevuto la presentazione ed è pronta a verificare:
  - In questo modo la privacy dello studente è stata preservata, in quanto solo le informazioni strettamente necessarie sono state condivise.

2.2.3 Protocollo di Verifica delle Credenziali

Questo protocollo descrive i passaggi che l'Università Verificatrice esegue per stabilire con certezza crittografica la validità e l'autenticità degli attributi presentati dallo studente.

- **Attori Coinvolti:** Università Verificatrice, Distributed Ledger Technology (DLT);
- **Pre-condizioni:** L'Università Verificatrice ha ricevuto il pacchetto di presentazione dallo studente e ha accesso alla DLT;
- **Step:**

1. **Analisi del Pacchetto Ricevuto:** per prima cosa, l'Università Verificatrice riceve il pacchetto di presentazione e ne estrae i singoli componenti: gli attributi condivisi dallo studente, la prova crittografica (*Proof of Membership*), la radice del Merkle Tree e la firma digitale originale dell'Università Emittente;
2. **Verifica dell'Autenticità (Firma Digitale):** il primo controllo consiste nel verificare la provenienza della credenziale. L'Università Verificatrice recupera la chiave pubblica dell'Università Emittente consultando la DLT e la utilizza per verificare la firma digitale apposta sulla radice del Merkle Tree. Se la firma non dovesse essere valida, significherebbe che la credenziale non è autentica e il processo si interromperebbe immediatamente;
3. **Verifica dell'Integrità (Prova Crittografica):** successivamente, si assicura che i dati siano integri, e quindi non siano stati manomessi in qualche modo. Utilizzando gli attributi e la prova crittografica forniti dallo studente, l'Università Verificatrice ricalcola l'hash della radice dell'albero. Il risultato di questo calcolo deve corrispondere esattamente alla radice del Merkle Tree ricevuta nel pacchetto. In caso contrario, i dati sono stati alterati e il processo di verifica fallisce;
4. **Controllo di Validità e Revoca su DLT:** come ultimo passaggio, si controlla lo stato attuale della credenziale. L'Università Verificatrice interroga la DLT per confermare che la radice del Merkle Tree sia stata ufficialmente registrata dall'Università Emittente e che non risulti in alcuna lista di revoca;
5. **Esito Finale:** la presentazione viene considerata valida solo se tutti i controlli precedenti sono andati a buon fine;

- **Post-condizioni:** L'Università Verificatrice ha stabilito la piena validità degli attributi presentati e può procedere con le azioni successive, come il riconoscimento degli esami.

2.2.4 Protocollo di Revoca delle Credenziali

Questo protocollo descrive il processo con cui l'Università Emittente può invalidare permanentemente una credenziale già emessa, il tutto in sicurezza e in maniera decentralizzata.

- **Attori Coinvolti:** Università Emittente, Distributed Ledger Technology (DLT);
- **Pre-condizioni:** l'Università Emittente ha motivo di revocare una credenziale. L'Università Emittente possiede la propria chiave privata, necessaria per autorizzare l'operazione in modo sicuro;
- **Step:**

1. **Identificazione della Credenziale:** l'Università Emittente identifica la credenziale da revocare tramite la radice del Merkle Tree, che funge da suo identificatore pubblico sul DLT;
2. **Creazione e Firma del Messaggio di Revoca:** per garantire che solo l'emittente originale possa autorizzare l'operazione, viene creato un messaggio di revoca. Questo messaggio viene firmato digitalmente dall'Università Emittente utilizzando la propria chiave privata;
3. **Pubblicazione su DLT:** il messaggio di revoca firmato viene inviato come transazione alla DLT, la quale lo registra in modo permanente e immutabile, associandolo alla radice dell'albero della credenziale;

- **Meccanismo di Verifica (lato Verificatore):** la verifica dello stato di revoca è un passaggio obbligatorio del protocollo di verifica (come già affermato nella sotto-sezione 2.2.3), garantendo che una credenziale revocata non possa essere utilizzata;
- **Post-condizioni:**
  - Lo stato della credenziale è registrato in modo permanente come "revocato" sulla DLT;
  - Qualsiasi tentativo futuro di verificare la credenziale avrà esito negativo durante il controllo sulla DLT.

Capitolo 3

WP 3: Analisi della sicurezza

Responsabile WP3:	Gabriele Imparato

Questa sezione analizza in dettaglio la robustezza dell'architettura proposta, valutando come le scelte tecnologiche descritte nel WP2 rispondano efficacemente alle minacce identificate nel WP1. L'analisi si fonda sui principi di crittografia e sicurezza informatica trattati nel corso, dimostrando che il sistema è progettato per garantire le proprietà di sicurezza fondamentali in un contesto avversariale.

3.1 Garanzie di Sicurezza Fondamentali

Il sistema è stato progettato per soddisfare le cinque proprietà cardine della sicurezza informatica moderna: *integrità*, *autenticità*, *privacy*, *non ripudio* e *revocabilità*.

3.1.1 Analisi: Integrità

L'integrità garantisce che i dati non siano stati alterati o manomessi.

La principale minaccia all'integrità proviene da uno studente malintenzionato o da un attaccante esterno che tenti di modificare i dati di una credenziale. La nostra architettura neutralizza questa minaccia attraverso un duplice meccanismo:

1. **Merkle Tree con impronta digitale**: ogni credenziale è strutturata come un Merkle Tree, dove la radice (Merkle Root o MTR) agisce come un'impronta digitale (fingerprint) dell'intera credenziale. Grazie alla resistenza alle collisioni della funzione di hash *SHA-256*, qualsiasi modifica a un singolo attributo altererebbe irrimediabilmente la MTR;

2. **Firma digitale sulla Merkle Root**: l'università emittente firma digitalmente la MTR applicando il paradigma "**hash-and-sign**". Utilizzando la propria chiave privata RSA, lega la propria identità a quell'impronta digitale. Qualsiasi manomissione invaliderebbe la firma, rendendo la frode immediatamente rilevabile in fase di verifica.

3.1.2 Analisi: Autenticità

L'autenticità garantisce la provenienza certa dei dati da un mittente dichiarato.

Le minacce all'autenticità includono la creazione di credenziali false o l'impersonificazione di un'università. Il sistema garantisce l'autenticità tramite:

- **Crittografia a chiave pubblica**: solo l'università emittente possiede la chiave privata necessaria per firmare la Merkle-Tree root. Qualsiasi verificatore può usare la chiave pubblica dell'università per controllare la firma, assicurandosi della sua provenienza;
- **DLT con Public Key Directory**: la DLT funge da registro pubblico e decentralizzato per le chiavi pubbliche dell'università, sostituendo la necessità di una Certification Authority (CA) centrale. Il verificatore recupera la chiave pubblica autentica direttamente dalla DLT.

3.1.3 Analisi: Privacy

La privacy garantisce che solo le entità autorizzate possano accedere alle informazioni.

Questa proprietà è minacciata da enti che raccolgono dati non necessari o da attaccanti che intercettano le comunicazioni. Le contromisure adottate sono:

1. **Divulgazione selettiva e principio del minimo privilegio**: grazie ai Merkle Tree, lo studente presenta solo gli attributi richiesti e la relativa **Merkle Proof**, applicando rigorosamente il *principio del minimo privilegio*;
    - Il principio del minimo privilegio dichiara che un utente o un processo dovrebbe avere solo i permessi strettamente necessari per svolgere il proprio compito, riducendo il rischio di attacchi;
2. **Minimizzazione dei dati su DLT**: la DLT non memorizza mai dati personali in chiaro, ma solo hash (le MTR), protetti dalla proprietà di **hiding** (**non-reversibilità**);
3. **Canali cifrati**: la comunicazione in transito avviene attraverso canali sicuri che implementano la cifratura autenticata, come **TLS** (**Transport Layer Security**), per proteggere i dati da intercettazioni.

3.1.4 Analisi: Non Ripudio

Il non ripudio impedisce a un mittente di negare di aver inviato un messaggio.

Una università emittente potrebbe tentare di negare di aver rilasciato una credenziale. Il sistema lo previene con:

- **Firma digitale pubblicamente verificabile**: la firma digitale sulla MTR costituisce una prova innegabile. Poiché la chiave pubblica è accessibile a tutti tramite la DLT, il ricevente (o una terza parte, come un giudice) può verificare autonomamente e in qualsiasi momento la validità e la provenienza della credenziale.

3.1.5 Analisi: Revocabilità

La revocabilità permette di invalidare una credenziale prima della sua naturale scadenza.

Questa proprietà è cruciale in caso di **compromissione del wallet** dello studente o di **emissione fraudolenta**. Il sistema gestisce la revoca in modo efficiente e sicuro:

- **Protocollo di revoca su DLT**: invece di affidarsi a **Certificate Revocation Lists** (**CRL**) tradizionali, il sistema registra le revoche direttamente sulla DLT . L'università emittente pubblica un messaggio di revoca, firmato con la sua chiave privata, che viene legato in modo immutabile alla MTR della credenziale;
- **Verifica obbligatoria**: il protocollo di verifica impone al ricevente di controllare lo stato della credenziale sulla DLT come ultimo passo, garantendo che una credenziale revocata non possa essere utilizzata.

3.2 Analisi vulnerabilità delle soluzioni proposte

3.2.1 Analisi: Studente che presenta credenziali false

- *Soluzione attuale*: Firma digitale + Verifica con DLT;
- *Analisi*: l'utilizzo della firma digitale garantisce l'autenticità e l'integrità delle credenziali emesse dall'università di partenza. La DLT funge da registro immutabile e condiviso per gli hash delle credenziali, permettendo al verificatore (Università di Salerno) di confrontare l'hash delle credenziali presentate con quello registrato, rilevando eventuali alterazioni. Questo riduce significativamente il rischio di falsificazione delle credenziali da parte dello studente;
- *Possibili miglioramenti*:
  - Adozione di standard comuni;
  - Verifica automatica lato università ricevente;

3.2.2 Analisi: Università negligente o malinentenzionata

- *Soluzione attuale*: tracciabilità della revoca con Distributed Ledger;
- *Analisi*: l'università ha accesso completo all'infrastruttura di ateneo. Il problema di un ente accademico che rilascia credenziali errate o incomplete può essere ridotto da una maggiore trasparenza e tracciabilità. L'uso di un DLT, per la gestione delle revoche, offre una tracciabilià delle azioni effettuate su di essa. Questo significa che tutte le possibili revoche sono registrate sulla DLT e verificabili in qualsiasi momento da chiunque, rendendo difficile per un ateneo negare una revoca o manipolare le credenziali da rilasciare;
- *Possibili miglioramenti*:
  - Linee guida condivise;
  - Verifica e ispezione formale tra enti;

3.2.3 Analisi: Attacchi esterni (terze parti)

- *Soluzione attuale*: Canali cifrati + Integrità con firma;
- *Analisi*: l'utilizzo di canali cifrati (es. TLS/SSL) per la comunicazione tra gli attori (università di partenza-studente, studente-università ospitante) è fondamentale per garantire la privacy e la riservatezza delle informazioni. La firma digitale assicura l'integrità dei messaggi, permettendo al destinatario di rilevare qualsiasi tentativo di alterazione durante la trasmissione. Questo contrasta efficacemente gli attacchi di *eavesdropper* e *man-in-the-middle*;
- *Possibili miglioramenti*:
  - Autenticazione forte degli attori;
  - Rilevamento di possibili anomalie;

3.2.4 Analisi: Violazione della privacy

- *Soluzione attuale*: Selettività dei dati + Minimizzazione dei dati;
- *Analisi*: la possibilità di presentare solo un sottoinsieme di campi della credenziale (presentazione selettiva, ad esempio con Merkle Tree) è cruciale per il rispetto del principio di minimizzazione dei dati da condividere. Questo consente allo studente di presentare solo le informazioni strettamente necessarie (es. riconoscimento esami) senza esporre dati sensibili non pertinenti (es. borsa di studio).
- *Possibili miglioramenti*:
  - Controllo più semplice per l'utente;
  - Policy chiare su dati sensibili.

3.3 Analisi della sicurezza scelte progettuali

3.3.1 Analisi Merkle Tree

L'impiego del Merkle Tree all'interno del sistema di gestione delle credenziali accademiche consente di ottenere un equilibrio tra integrità, efficienza e riservatezza dei dati. Ogni credenziale viene rappresentata come una struttura ad albero binario in cui ogni foglia è costituita dall'hash di (key, value). I nodi interni dell'albero sono generati combinando gli hash delle foglie sottostanti fino a raggiungere la radice (*Merkle root*).

Dal punto di vista della sicurezza, questo approccio offre i seguenti vantaggi:

- **Integrità**: ogni modifica a un singolo attributo della credenziale cambia l'intera Merkle root, rendendo immediatamente rilevabili eventuali alterazioni.
- **Selective Disclosure**: lo studente può dimostrare la validità di singoli attributi (es. media voti) senza rivelare l'intero contenuto della credenziale, fornendo un *Merkle proof* e il valore hashato.
- **Efficienza**: la verifica di un singolo campo richiede solo una piccola parte dell'albero, garantendo tempi rapidi di validazione.

Il Merkle Tree è anche compatibile con una logica di *privacy-by-design*, in quanto consente a chi verificare di ottenere garanzie crittografiche senza accedere direttamente a dati.[2]

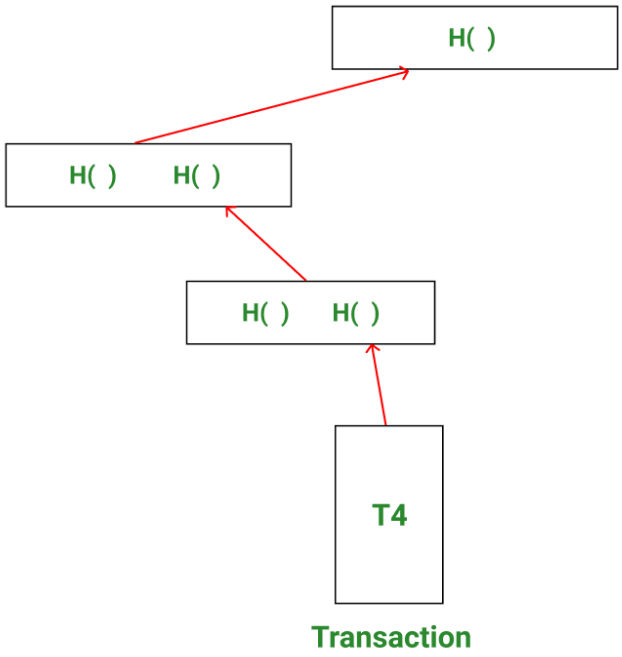

Figura 3.1: Esempio di proof of membership per una transazione.

Dal punto di vista crittografico, la sicurezza del Merkle Tree si basa sull'utilizzo di funzioni hash resistenti alle collisioni (es. SHA-256), che garantiscono che due foglie diverse non generino lo stesso hash. Ciò rende altamente improbabile che un attore possa costruire un proof fraudolento con attributi falsificati.

3.3.2 Analisi DLT

La DLT ha il compito di garantire un ledger immutabile e distribuito per il tracciamento delle credenziali accademiche. In particolare, il registro conserva:

- Le radici delle credenziali emesse;
- Le informazioni relative alla revoca delle credenziali;
- Eventuali *timestamp* associati alle transazioni.

Dal punto di vista della sicurezza, la DLT garantisce:

- **Immutabilità**: le informazioni registrate non possono essere modificate retroattivamente, fornendo una prova di esistenza e integrità nel tempo.
- **Disponibilità**: essendo distribuito, il ledger è resiliente a guasti di singoli nodi o attacchi mirati.
- **Trasparenza e verificabilità**: ogni attore può consultare lo stato di una credenziale o verificare la firma di un emittente senza dover contattare direttamente una terza parte.
- **Decentralizzazione**: il sistema evita la concentrazione del potere in un'entità centrale, riducendo i rischi sistemici e aumentando la fiducia complessiva.

L'utilizzo della DLT è compatibile con l'approccio trustless e interoperabile.

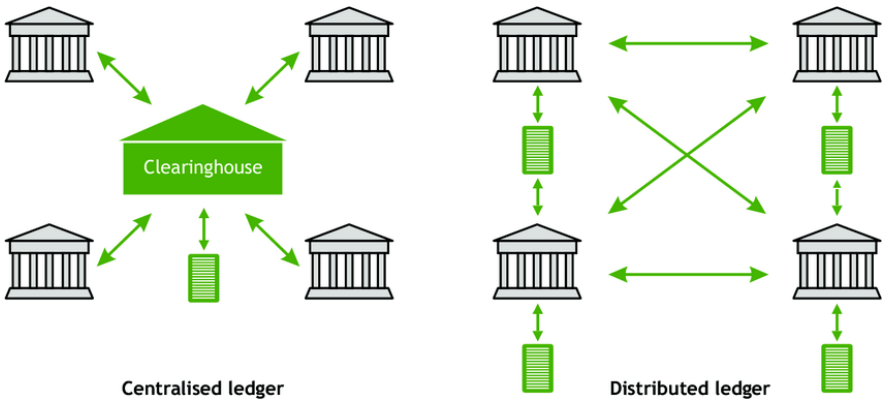

Figura 3.2: Struttura di un DLT centralizzato e decentralizzato.

Differenza tra DLT centralizzato e non centralizzato:

- **DLT centralizzato**: è gestito da un unico ente o gruppo di enti che controllano il registro. Anche se è distribuito tecnicamente, c'è un punto di controllo e di fiducia unico, quindi meno trasparente e più vulnerabile a censura o guasti;
- **DLT decentralizzato**: nessun singolo ente ha il controllo totale; il registro è distribuito su tanti nodi indipendenti che validano l'insieme le transazioni. Questo garantisce maggiore sicurezza, trasparenza e resistenza a manipolazioni o attacchi.

Per garantire sicurezza e trasparenza, in questo task come in molti altri si preferisce adottare una struttura decentralizzata, che aumenta la resistenza contro tentativi di manipolazione e rende il sistema più robusto agli attacchi.

In una versione avanzata del sistema, si può adottare una DLT permissioned (es. Hyperledger Fabric o Indy), che consente di bilanciare trasparenza e riservatezza, limitando l'accesso alla scrittura su registro a un insieme di enti accreditati. Questo approccio è particolarmente adatto in ambito accademico, dove è necessario identificare chiaramente i soggetti emettitori.[3]

3.3.3 Analisi Hardware Wallet

Il wallet fisico è un componente cruciale per la protezione dell'identità e delle credenziali dello studente. Può essere realizzato sotto forma di **smart card**, **chiavetta USB** con algoritmo di cifratura installato oppure **dispositivo NFC** sicuro. Esso ha il compito di custodire:

- Le chiavi private necessarie per la firma delle credenziali o dei Merkle proof;
- Le credenziali digitali ricevute dalle università (in forma firmata e hashata).

Dal punto di vista della sicurezza, il wallet fisico offre:

- **Sicurezza locale**: la chiave privata non lascia mai il dispositivo, riducendo il rischio di compromissione da remoto.
- **Controllo personale**: lo studente è l'unico titolare dell'accesso al wallet, evitando la necessità di server centralizzati o intermediari fidati.
- **Portabilità e privacy**: la credenziale può essere trasportata ed esibita offline, senza obbligo di connessione o tracciamento.
- **Resistenza agli attacchi**: i dispositivi hardware più avanzati offrono protezione da attacchi fisici o digitali.

L'impiego del wallet fisico è considerato sicuro in quanto permette allo studente di controllare direttamente la diffusione dei propri dati personali.

Figura 3.3: Esempi di wallet hardware tra cui: una smart card, una usb sicura e un NFC.

Per aumentare ulteriormente la robustezza del wallet fisico, si potrebbe prevedere l'uso di meccanismi di backup cifrato e protezione tramite PIN o autenticazione biometrica. In caso di smarrimento o furto del dispositivo, tali misure limitano i rischi di compromissione delle credenziali.[1]

3.4 Analisi di Rischi Residui e Contromisure Avanzate

Anche un'architettura robusta presenta potenziali vettori di attacco che richiedono un'analisi più approfondita.

3.4.1 Replay Attack

- **Scenario di attacco**: un attaccante intercetta una presentazione valida e la ripresenta in un secondo momento;
- **Vulnerabilità**: sebbene i dati siano integri e autentici, il sistema verificatore potrebbe processare una richiesta legittima più volte;
- **Contromisura suggerita**: integrare un meccanismo di **challenge-response**. L'università verificatrice invia un valore casuale monouso (**nonce**) che lo studente deve includere nella presentazione firmata, rendendo ogni richiesta unica.

3.4.2 Timing Attack

- **Scenario di attacco**: un attaccante con accesso al wallet fisico misura il tempo di esecuzione delle operazioni crittografiche per dedurre informazioni sulla chiave privata;
- **Vulnerabilità**: se le operazioni di firma non sono implementate in tempo costante, potrebbero rivelare dati segreti;
- **Contromisura suggerita**: questa vulnerabilità è mitigata a livello implementativo. Utilizzando una libreria crittografica moderna come `cryptography.hazmat`, si sfrutta una base di codice già progettata per resistere a noti attacchi side-channel.

3.4.3 Man-in-the-Middle in Fase di Registrazione

- **Scenario di attacco**: durante la registrazione iniziale della sua chiave pubblica sulla DLT, un'università subisce un attacco Man-in-the-Middle che sostituisce la sua chiave legittima con una dell'attaccante;
- **Vulnerabilità**: questo è il "problema del bootstrap" della fiducia. Se una chiave pubblica falsa viene accettata come autentica, l'intero modello di sicurezza crolla;
- **Contromisura suggerita**: questo non è un difetto del protocollo transazionale, ma della fase di accreditamento iniziale dell'ente. La registrazione e la validazione di una nuova università sulla DLT devono essere protette da un processo di autenticazione forte e "out-of-band", ad esempio tramite la validazione da parte di altri membri già fidati della rete o attraverso canali istituzionali verificati. Tale procedura è concettualmente simile a quella con cui opera una Public-Key Authority nel mondo centralizzato per associare un'identità a una chiave pubblica .

3.5 Garanzie di Sicurezza e Privacy

3.5.1 Garanzie di Privacy

La protezione dei dati personali dello studente è un obiettivo fondamentale, ottenuta attraverso:

- **Divulgazione Selettiva (Selective Disclosure):** grazie ai Merkle Tree, lo studente rivela solo le informazioni necessarie, aderendo al principio di minimizzazione dei dati;
- **Minimizzazione dei Dati sulla DLT:** sul registro distribuito vengono memorizzati solo hash e metadati, mai dati personali, proteggendo l'identità dello studente grazie alla non-reversibilità delle funzioni hash;

- **Applicazione del Principio di Minimo Privilegio:** in linea con il principio di *Least Privilege*, l'architettura garantisce che sia lo studente ad avere il controllo completo sulla condivisione dei propri dati. Il controllo viene esercitato tramite il wallet personale, che mantiene e gestisce le chiavi necessarie a generare la presentazione selettiva. Ogni condivisione di dati, pertanto, richiede un'azione esplicita e autorizzata solo dallo studente.

3.5.2 Garanzie di Sicurezza

Le proprietà di sicurezza fondamentali sono garantite da:

- **Autenticità:** la firma digitale RSA garantisce che ogni credenziale provenga dall'emittente dichiarato;
- **Integrità:** la firma sulla radice del Merkle Tree e l'immutabilità della DLT assicurano che i dati non possano essere manomessi dopo l'emissione;
- **Non Ripudio (per l'Emittente):** la presenza di firma e registrazione su DLT crea una prova innegabile dell'emissione;
- **Resistenza alla Manomissione:** falsificare una credenziale è computazionalmente infattibile se non si ha a disposizione la chiave privata dell'emittente.

3.5.3 Gestione della Revoca

Il meccanismo di revoca è robusto grazie a:

- **Tempestività ed Efficacia:** il DLT propaga lo stato di revoca in modo rapido ed efficiente a tutti i partecipanti della rete;
- **Autenticazione della Revoca:** solo l'emittente originale può autorizzare una revoca, firmando la richiesta con la propria chiave privata.

3.5.4 Decentralizzazione e Resistenza alla Censura

L'uso della DLT elimina la dipendenza da un ente centrale, offrendo:

- **Eliminazione dei Single Point of Failure:** il sistema rimane operativo anche in caso di guasto di alcuni nodi;
- **Resistenza alla Censura:** nessuna singola entità può impedire l'emissione o la verifica di credenziali legittime.

3.5.5 Interoperabilità

La soluzione promuove l'interoperabilità attraverso l'adozione di standard aperti per i formati dei dati (JSON), i protocolli crittografici (SHA-256, RSA) e le interfacce con la DLT.

3.5.6 Analisi del Threat Model e Contromisure

La soluzione mitiga le minacce che abbiamo identificato all'interno del WP1:

- **Studente che falsifica credenziali:** ciò risulta impossibile senza la firma dell'emittente;
- **Attaccante Man-in-the-Middle:** questa ipotesi è inefficace, poiché qualsiasi modifica renderebbe invalida la firma digitale;
- **Compromissione della chiave dell'emittente:** richiede un processo di revoca della chiave e l'emissione di una nuova coppia di chiavi;
- **Compromissione del wallet studente:** lo studente dovrebbe tempestivamente notificarlo all'emittente, che può così procedere a revocare le credenziali compromesse sulla DLT.

3.6 Sintesi dell'architettura sicura

L'architettura proposta adotta un approccio multi-strato alla sicurezza. L'uso combinato di primitive crittografiche standard (firme RSA-PSS, hash SHA-256), strutture dati efficienti (Merkle Tree) e un modello di fiducia decentralizzato (DLT) fornisce una solida difesa contro le minacce identificate. I rischi residui, come i replay attacks, sono noti e possono essere mitigati con estensioni al protocollo, garantendo un sistema complessivamente sicuro, privato e affidabile.

Capitolo 4

WP 4: Implementazione e prestazioni

Responsabile WP4:	Michele Penna

4.1 Strumenti e librerie utilizzate

Per simulare il comportamento del sistema sono stati individuati alcuni strumenti open-source e librerie comunemente utilizzate in ambito DLT, crittografia e gestione delle identità digitali. La selezione è stata fatta tenendo conto della compatibilità con il modello progettuale, della facilità di utilizzo in ambienti accademici e della disponibilità di documentazione.

- **Linguaggio Python**: utilizzato come linguaggio di scripting per simulare i flussi principali del sistema (generazione credenziali, creazione Merkle Tree, firma digitale, verifica);
- **Libreria cryptography**: utilizzata per la generazione di chiavi pubbliche/private, la firma digitale e la verifica delle firme. Garantisce l'integrità delle credenziali durante la trasmissione tra attori;
- **Libreria hashlib**: impiegata per l'hashing degli attributi e la costruzione delle radici di Merkle Tree, fondamentale per la validazione selettiva delle credenziali;
- **Libreria os**: utilizzata per la generazione di nonce casuali in fase di challenge-response e per la gestione di operazioni base sul file system;
- **Interfaccia wallet (mock)**: rappresentata da uno script Python che consente allo studente di conservare le credenziali emesse da una o più università.

4.2 Implementazione della soluzione

4.2.1 Costruzione del Merkle Tree

La classe `MerkleTree` implementa una struttura dati crittografica ad albero, fondamentale per la gestione efficiente e sicura delle prove di integrità e appartenenza. Attraverso la costruzione dell'albero a partire da foglie rappresentanti gli attributi della credenziale, essa consente di:

- **Calcolare la radice Merkle:** una singola impronta crittografica che sintetizza in modo univoco l'intero insieme degli attributi, usata come riferimento per la firma digitale.
- **Verificare le prove di appartenenza (Merkle proofs):** dati uno o più attributi, è possibile dimostrare che essi fanno parte dell'insieme originario senza dover rivelare tutte le altre informazioni, garantendo così selettività e privacy.

Questa struttura è essenziale per consentire la verifica efficiente e sicura delle credenziali in ambienti distribuiti, minimizzando l'esposizione dei dati sensibili.

```python
from cryptography.hazmat.primitives import hashes

class MerkleTree:
    """
    Implementazione di un Merkle Tree.
    """
    def __init__(self, data_list: list):
        self.leaves = [self._hash_function(data) for data in data_list]
        self.tree = [self.leaves]
        self._build_tree()

    @staticmethod
    def _hash_function(data: bytes) -> str:
        digest = hashes.Hash(hashes.SHA256())
        digest.update(data)
        return digest.finalize().hex()

    def _build_tree(self):
        level = list(self.leaves)
        if not level:
            return
        while len(level) > 1:
            if len(level) % 2 != 0:
                level.append(level[-1])
            next_level = []
            for i in range(0, len(level), 2):
                parent_data = (level[i] + level[i+1]).encode('utf-8')
                next_level.append(self._hash_function(parent_data))
            self.tree.append(next_level)
            level = next_level

    def get_root(self) -> str:
        return self.tree[-1][0] if self.tree and self.tree[-1] else None
```

```
34
35      def get_proof(self, leaf_index: int) -> list:
36          proof = []
37          current_index = leaf_index
38          for level in self.tree[:-1]:
39              is_right_node = current_index % 2 != 0
40              sibling_index = current_index - 1 if is_right_node else current_index +
     1
41              if sibling_index < len(level):
42                  proof.append({"hash": level[sibling_index], "position": "left" if
    is_right_node else "right"})
43              current_index //= 2
44          return proof
45
46      @staticmethod
47      def verify_proof(root: str, leaf_data: bytes, proof: list) -> bool:
48          computed_hash = MerkleTree._hash_function(leaf_data)
49          for proof_element in proof:
50              if proof_element["position"] == 'right':
51                  combined_data = (computed_hash + proof_element["hash"]).encode('utf
    -8')
52              else:
53                  combined_data = (proof_element["hash"] + computed_hash).encode('utf
    -8')
54              computed_hash = MerkleTree._hash_function(combined_data)
55          return computed_hash == root
```

Listing 4.1: Implementazione del Merkle Tree

4.2.2 Costruzione del DLT simulato

La classe `DLT` simula una Distributed Ledger Technology, ovvero un registro distribuito e immutabile, utilizzato per mantenere la tracciabilità e la trasparenza del sistema di gestione delle credenziali digitali. Le funzionalità principali includono:

- **Registrazione delle chiavi pubbliche:** memorizza le chiavi pubbliche delle università partecipanti, garantendo la possibilità di verificare firme digitali in modo affidabile e decentralizzato.
- **Gestione delle credenziali emesse:** tiene traccia delle credenziali registrate, incluse informazioni sul loro stato, come eventuali revoche, in modo da prevenire l'uso fraudolento o non autorizzato.

La DLT funge da infrastruttura di fiducia distribuita, essenziale per supportare un sistema di identità digitale resistente a manomissioni, censura e attacchi esterni.

```
1 class DLT:
2     """ Simulation of a Distributed Ledger Technology. """
3     def __init__(self):
4         self.universities_pks = {}
5         self.merkle_roots = set()
6         self.revocation_list = {}
7
8     def register_university_pk(self, uni_name: str, public_key):
```

```
        print(f"[DLT] Registered public key for: {uni_name}")
        self.universities_pks[uni_name] = public_key

    def get_university_pk(self, uni_name: str):
        return self.universities_pks.get(uni_name)

    def register_credential(self, merkle_root: str):
        print(f"[DLT] Registered new Merkle Root: {merkle_root[:10]}...")
        self.merkle_roots.add(merkle_root)

    def revoke_credential(self, merkle_root: str, revocation_signature):
        if merkle_root in self.merkle_roots: self.revocation_list[merkle_root] =
    revocation_signature

    def get_credential_status(self, merkle_root: str):
        if merkle_root in self.revocation_list: return "REVOKED"
        if merkle_root in self.merkle_roots: return "VALID"
        return "NOT REGISTERED"
```

Listing 4.2: Implementazione del DLT simulato

4.2.3 Costruzione dell'attore: Università

La classe `University` rappresenta un'entità accademica responsabile dell'emissione, gestione e revoca delle credenziali digitali degli studenti. Ogni istanza di questa classe è associata a una coppia di chiavi crittografiche (pubblica e privata) utilizzata per firmare digitalmente le credenziali, garantendone autenticità e integrità. La classe offre metodi per:

- **Emissione della credenziale:** costruisce un Merkle Tree a partire dagli attributi selezionati dello studente, calcolandone la radice e firmandola digitalmente con la chiave privata. Questo processo consente di garantire una prova crittografica dell'appartenenza degli attributi senza esporre l'intero set di dati.
- **Verifica delle presentazioni:** controlla che la presentazione ricevuta dallo studente contenga una firma valida, che la prova Merkle associata sia corretta e che il nonce usato nella sfida sia quello previsto, in modo da prevenire attacchi di replay.
- **Revoca delle credenziali:** registra lo stato di revoca di una specifica credenziale nel registro distribuito (DLT), rendendo inefficace qualsiasi successiva presentazione di quella credenziale.

Questa classe incarna quindi il ruolo di emittente fiduciario nel sistema di gestione delle identità digitali basate su DLT e Merkle Tree.

```
from MerkleTree import MerkleTree
from DLT import DLT
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes
from cryptography.exceptions import InvalidSignature
import json
```

```
7
8  class University:
9      """ Represents a university issuing credentials. """
10     def __init__(self, name):
11         self.name = name
12         self.private_key = rsa.generate_private_key(public_exponent=65537, key_size
       =2048)
13         self.public_key = self.private_key.public_key()
14         self.used_nonces = set()
15
16     def get_public_key(self):
17         return self.public_key
18
19     def issue_credential(self, student_attributes: dict):
20         print(f"\n[ISSUING] {self.name} is issuing a credential...")
21         leaves_data = [json.dumps({k: v}, sort_keys=True).encode('utf-8') for k, v
       in student_attributes.items()]
22         tree = MerkleTree(leaves_data)
23         merkle_root = tree.get_root()
24         signature = self.private_key.sign(
25             merkle_root.encode('utf-8'),
26             padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=padding.PSS.
       MAX_LENGTH),
27             hashes.SHA256()
28         )
29         credential_package = {"attributes": student_attributes, "merkle_root":
       merkle_root, "signature": signature}
30         print(f"[ISSUING] Success for issuing credential to {student_attributes.get
       ('name')}.")
31         return credential_package
32
33     def verify_presentation(self, presentation: dict, dlt: 'DLT', expected_nonce:
       str):
34         print(f"\n[VERIFICATION] {self.name} is verifying a presentation...")
35
36         if not presentation:
37             print("  - ERROR: Presentation is None. Cannot verify.")
38             return False
39
40         try:
41             receved_nonce = presentation.get("nonce")
42             if not receved_nonce:
43                 raise ValueError("Missing nonce in presentation.")
44             if receved_nonce in self.used_nonces:
45                 raise ValueError("Replay Attack Detected! Nonce already used.")
46             if receved_nonce != expected_nonce:
47                 raise ValueError("Invalid nonce. The request may be outdated or
       malicious.")
48             print(f"  - STEP 0/4: Nonce '{receved_nonce[:8]}...' valid and never
       used before.")
49
50             receved_merkle_root = presentation["merkle_root"]
51             issuer_pk = dlt.get_university_pk(presentation["issuer"])
52             issuer_pk.verify(
53                 presentation["signature"],
54                 receved_merkle_root.encode('utf-8'),
```

```
                padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=padding.PSS.MAX_LENGTH),
                hashes.SHA256()
            )
            print("  - STEP 1/4: Digital Signature AUTHENTIC.")

            is_proof_valid = MerkleTree.verify_proof(
                receved_merkle_root,
                json.dumps(presentation["selected_attribute"], sort_keys=True).encode('utf-8'),
                presentation["proof_of_membership"]
            )
            if not is_proof_valid:
                raise ValueError("Merkle Proof not valid. Data potentially tampered.")
            print("  - STEP 2/4: Merkle Proof valid. Data INTACT.")

            status = dlt.get_credential_status(receved_merkle_root)
            if status != "VALID":
                raise ValueError(f"Status of credential on DLT not valid: {status}.")
            print("  - STEP 3/4: Credential VALID and NOT REVOKED on DLT.")

            self.used_nonces.add(receved_nonce)
            print(f"  - STEP 4/4: Nonce '{receved_nonce[:8]}...' registered as used.")
            print(f"[VERIFICATION] Presentation verified successfully.")
            return True

        except (InvalidSignature, ValueError) as e:
            print(f"  - VERIFICATION ERROR: {e}")
            return False
        except Exception as e:
            print(f"  - UNKNOWN ERROR during verification: {e}")
            return False


    def revoke_credential(self, merkle_root: str, dlt: 'DLT'):
        print(f"\n[REVOKE] {self.name} is revoking credential with MTR: {merkle_root[:10]}...")
        revoke_message = f"REVOKE:{merkle_root}".encode('utf-8')
        revoke_sign = self.private_key.sign(revoke_message, padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=padding.PSS.MAX_LENGTH), hashes.SHA256())
        dlt.revoke_credential(merkle_root, revoke_sign)
        print("[REVOKE] Credential revoked successfully on DLT.")
```

Listing 4.3: Implementazione dell'attore Università

4.2.4 Costruzione dell'attore: Studente

La classe `Student` rappresenta il soggetto titolare delle credenziali digitali emesse dalle università. Il wallet digitale associato a ciascun oggetto `Student` funge da archivio locale delle credenziali ricevute, organizzate per università emittenti. Le funzionalità principali includono:

- **Generazione di presentazioni selettive:** consente allo studente di esporre solo un sottoinsieme degli attributi contenuti nella credenziale, accompagnandoli con le relative prove Merkle che attestano la loro validità senza rivelare dati aggiuntivi.
- **Gestione della revoca:** permette la rimozione delle credenziali revocate dal wallet, impedendo così l'uso non autorizzato o non valido in future interazioni.

In sintesi, la classe `Student` funge da portatore e controllore delle proprie informazioni digitali, garantendo al contempo la privacy e la sicurezza nel processo di presentazione delle credenziali.

```python
from MerkleTree import MerkleTree
import json

class Student:
    """ Represents a student and their physical wallet. """
    def __init__(self, name, wallet):
        self.name = name
        self.wallet = wallet  # ora wallet è un'istanza di PhysicalWallet

    def revoke_credentials(self, issuer_name: str, credential_package: dict):
        print(f"\n[WALLET] {self.name} revoked credential for {issuer_name}.")
        self.wallet.delete_credential(issuer_name)

    def generate_presentation(self, issuer_name: str, presentation_attribute_key: str, nonce: str):
        print(f"\n[PRESENTATION] {self.name} is creating a selective presentation ...")
        credential_package = self.wallet.retrieve_credential(issuer_name)
        if not credential_package:
            print("  - ERROR: No credential found in wallet for the given issuer.")
            return None

        original_attributes = credential_package["attributes"]
        leaves_data = [json.dumps({k: v}, sort_keys=True).encode('utf-8') for k, v in original_attributes.items()]
        leaf_index = list(original_attributes.keys()).index(presentation_attribute_key)
        tree = MerkleTree(leaves_data)

        presentation = {
            "issuer": issuer_name,
            "selected_attribute": {presentation_attribute_key: original_attributes[presentation_attribute_key]},
            "proof_of_membership": tree.get_proof(leaf_index),
            "merkle_root": credential_package["merkle_root"],
            "signature": credential_package["signature"],
            "nonce": nonce
        }
        print(f"[PRESENTATION] Created presentation for '{presentation_attribute_key}' with nonce '{nonce[:8]}...'.")
        return presentation
```

Listing 4.4: Implementazione dell'attore Studente

4.2.5 Costruzione del wallet fisico

La classe `Student` rappresenta il soggetto titolare delle credenziali digitali emesse dalle università. Il wallet digitale associato a ciascun oggetto `Student` funge da archivio locale delle credenziali ricevute, organizzate per università emittenti. Le funzionalità principali includono:

- **Generazione di presentazioni selettive:** consente allo studente di esporre solo un sottoinsieme degli attributi contenuti nella credenziale, accompagnandoli con le relative prove Merkle che attestano la loro validità senza rivelare dati aggiuntivi.
- **Gestione della revoca:** permette la rimozione delle credenziali revocate dal wallet, impedendo così l'uso non autorizzato o non valido in future interazioni.

In sintesi, la classe `Student` funge da portatore e controllore delle proprie informazioni digitali, garantendo al contempo la privacy e la sicurezza nel processo di presentazione delle credenziali.+

```python
import json
import os

class PhysicalWallet:
    def __init__(self, owner_name):
        self.owner = owner_name
        self.credentials = {}  # {university_name: credential_data}

    def store_credential(self, issuer, credential):
        self.credentials[issuer] = credential

    def retrieve_credential(self, issuer):
        return self.credentials.get(issuer)

    def delete_credential(self, issuer):
        if issuer in self.credentials:
            del self.credentials[issuer]

    def save_to_disk(self, path):
        file_path = os.path.join(path, f"{self.owner}_wallet.json")

        # Conversione di eventuali valori bytes in stringa hex per il salvataggio
        serializable_credentials = {}
        for issuer, data in self.credentials.items():
            serializable_data = {}
            for k, v in data.items():
                if isinstance(v, bytes):
                    serializable_data[k] = v.hex()
                else:
                    serializable_data[k] = v
            serializable_credentials[issuer] = serializable_data

        with open(file_path, 'w') as f:
            json.dump(serializable_credentials, f, indent=4)

```

```
    def load_from_disk(self, path):
        file_path = os.path.join(path, f"{self.owner}_wallet.json")
        if os.path.exists(file_path):
            with open(file_path, 'r') as f:
                loaded_credentials = json.load(f)

            # Riconverte le firme da hex a bytes
            for issuer, data in loaded_credentials.items():
                if 'signature' in data and isinstance(data['signature'], str):
                    data['signature'] = bytes.fromhex(data['signature'])
            self.credentials = loaded_credentials
        else:
            raise FileNotFoundError(f"No wallet file found at {file_path}")
```

Listing 4.5: Implementazione del wallet fisico

4.2.6 Flusso di esecuzione e utilizzo

Il codice presentato simula l'emissione di una credenziale accademica digitale selettiva, utilizzando firme digitali, Merkle Tree e Distributed Ledger Technology (DLT). I passaggi di inizializzazione sono:

1. **Setup:** viene creato un ambiente simulato con due università (*Université de Rennes* e *Università di Salerno*), uno studente e un DLT;

2. **Registrazione chiavi:** le università registrano le proprie chiavi pubbliche nel DLT per permettere la verifica delle firme.

```
import os
import time
import json
from University import University
from Student import Student
from DLT import DLT
from PhysicalWallet import PhysicalWallet

# --- Environment Setup with Swapped Roles ---
dlt = DLT()
issuer_university = University("University of Salerno")
verifier_university = University("University of Rennes")
student_name = "Luigi Veniero"
wallet = PhysicalWallet(student_name)
student = Student(student_name, wallet)

# Register university public keys on the DLT
dlt.register_university_pk(issuer_university.name, issuer_university.get_public_key
    ())
dlt.register_university_pk(verifier_university.name, verifier_university.
    get_public_key())
```

Listing 4.6: Inizializzazione università, studente, wallet di prova e DLT.

Emissione della credenziale digitale

In questa fase, l'università emittente genera una credenziale digitale contenente informazioni accademiche dello studente. Il codice esegue i seguenti passaggi:

- Viene definito il dizionario `student_data`, che contiene attributi personali e accademici dello studente, come l'identificativo, il nome, il corso di laurea, il libretto con gli esami e il possesso o meno di una borsa di studio.
- L'università emittente (`issuer_university`) emette la credenziale tramite la funzione `issue_credential()`, che calcola un albero di Merkle con gli attributi forniti e firma la radice con la propria chiave privata. Il tempo di esecuzione viene misurato in millisecondi.
- La radice dell'albero (`merkle_root`) viene registrata sulla DLT per rendere la credenziale pubblicamente verificabile.
- La credenziale completa viene salvata all'interno del wallet digitale dello studente, associandola all'università emittente.
- Infine, viene stampato a video un messaggio che conferma l'archiviazione della credenziale nel wallet.

Questa fase costituisce il processo di onboarding iniziale dello studente, rendendo disponibili le informazioni accademiche in formato digitale, firmato e verificabile.

```
# --- 1. FULL CREDENTIAL ISSUANCE ---
student_data = {
    "student_ID": "N4600xxxx",
    "first_name": "Luigi",
    "last_name": "Veniero",
    "degree": "Computer Engineering Master's Degree",
    "exam_transcripts": {
        "Software Architecture Design": "28",
        "Data Science": "27",
        "Algoritmi e Protocolli per la Sicurezza": "18"
    },
    "citizenship": "Italian",
    "scholarship": True
}

start_issuance = time.perf_counter()
credential = issuer_university.issue_credential(student_data)
issuance_time = (time.perf_counter() - start_issuance) * 1000

dlt.register_credential(credential["merkle_root"])
student.wallet.store_credential(issuer_university.name, credential)
print(f"\n[WALLET] Credential from {issuer_university.name} stored in {student.name
    }'s wallet.")
```

Listing 4.7: Simulazione della consegna delle credenziali

```
[DLT] Registered public key for: University of Salerno
[DLT] Registered public key for: University of Rennes

[ISSUING] University of Salerno is issuing a credential...
[ISSUING] Success for issuing credential to None.
[DLT] Registered new Merkle Root: e53b5537c4...

[WALLET] Credential from University of Salerno stored in Luigi Veniero's wallet.
```

Figura 4.1: Output registrazione università su DLT ed emissione delle credenziali.

Flusso di verifica legittima (con nonce)

Questa fase simula una richiesta di verifica da parte di un'università ospitante (verificatore), in cui lo studente presenta un'informazione selezionata dalla sua credenziale, dimostrandone l'autenticità tramite una proof crittografica. Il codice esegue i seguenti passaggi:

- L'attributo da presentare viene scelto (`"degree"`), ad esempio per dimostrare il titolo di studio conseguito;
- L'università verificatrice genera un valore casuale (`nonce`), usato per prevenire attacchi di tipo replay, e lo invia allo studente come sfida crittografica;
- Lo studente genera una *presentazione selettiva* con il solo attributo richiesto, includendo:
  - la proof di validità (es. Merkle proof);
  - la firma calcolata sull'attributo e sul nonce ricevuto;

  Il tempo di generazione viene misurato;
- Il verificatore controlla che:
  - l'attributo sia parte della credenziale emessa (via la Merkle proof);
  - il nonce sia quello atteso;
  - la firma sia valida e riferita a una radice registrata sulla DLT;

  Anche il tempo di verifica viene misurato;
- Infine, viene stampato l'esito della verifica (`VALID` o `INVALID`).

Questa procedura dimostra l'utilizzo sicuro delle credenziali digitali in contesti di verifica decentralizzata e anti-falsificazione.

```python
# --- 2. LEGITIMATE VERIFICATION FLOW (with Nonce) ---
print("\n\n--- 2. LEGITIMATE VERIFICATION FLOW (with Nonce) ---")
attribute = "degree"
nonce_1 = os.urandom(16).hex()
print(f"[CHALLENGE] {verifier_university.name} sends a nonce to {student.name}: {
    nonce_1[:8]}...")

```

```
start_presentation = time.perf_counter()
presentation = student.generate_presentation(issuer_university.name, attribute,
    nonce=nonce_1)
presentation_time = (time.perf_counter() - start_presentation) * 1000

start_verification = time.perf_counter()
verification_result = verifier_university.verify_presentation(presentation, dlt,
    expected_nonce=nonce_1)
verification_time = (time.perf_counter() - start_verification) * 1000

print(f"\n---> First verification result: {'VALID' if verification_result else '
    INVALID'}")
```

Listing 4.8: Simulazione di verifica legittima delle credenziali usando un nonse

```
--- 2. LEGITIMATE VERIFICATION FLOW (with Nonce) ---
[CHALLENGE] University of Rennes sends a nonce to Luigi Veniero: 664fb052...

[PRESENTATION] Luigi Veniero is creating a selective presentation...
[PRESENTATION] Created presentation for 'degree' with nonce '664fb052...'.

[VERIFICATION] University of Rennes is verifying a presentation...
  - STEP 0/4: Nonce '664fb052...' valid and never used before.
  - STEP 1/4: Digital Signature AUTHENTIC.
  - STEP 2/4: Merkle Proof valid. Data INTACT.
  - STEP 3/4: Credential VALID and NOT REVOKED on DLT.
  - STEP 4/4: Nonce '664fb052...' registered as used.
[VERIFICATION] Presentation verified successfully.

---> First verification result: VALID
```

Figura 4.2: Output verifica legittima delle credenziali.

Simulazione di attacco di replay

In questa fase si simula un attacco di tipo **replay**, in cui un attaccante malintenzionato intercetta una presentazione legittima precedentemente inviata dallo studente e cerca di riutilizzarla senza modifiche, nel tentativo di farsi passare per lo studente.

- L'attaccante riutilizza la stessa presentazione generata nella fase precedente, contenente:
  - lo stesso attributo selezionato;
  - la stessa firma;
  - lo stesso nonce;
- Il verificatore riceve questa presentazione e tenta di validarla usando ancora il nonce precedente (`nonce_1`);

- Il risultato della verifica viene stampato: in un sistema ben progettato, questa seconda presentazione identica dovrebbe essere rifiutata, poiché il nonce è già stato utilizzato.

Questo tipo di test serve a verificare che il sistema sia protetto contro gli attacchi di riutilizzo, sfruttando il concetto di **one-time challenge** tramite `nonce`.

```
# --- 3. REPLAY ATTACK SIMULATION ---
print("\n\n--- 3. REPLAY ATTACK SIMULATION ---")
print("[REPLAY ATTACK] An attacker reuses a previously valid presentation...")
replay_result = verifier_university.verify_presentation(presentation, dlt,
    expected_nonce=nonce_1)
print(f"\n---> Replay attack result: {'VALID' if replay_result else 'INVALID'}")
```

Listing 4.9: Utilizzo in condizioni di replay attack

```
--- 3. REPLAY ATTACK SIMULATION ---
[REPLAY ATTACK] An attacker reuses a previously valid presentation...

[VERIFICATION] University of Rennes is verifying a presentation...
  - VERIFICATION ERROR: Replay Attack Detected! Nonce already used.

---> Replay attack result: INVALID
```

Figura 4.3: Output attacco di replay: nonce già usato

Simulazione di attacco per manipolazione dei dati

In questa fase viene simulato un attacco in cui un attaccante modifica direttamente i dati all'interno di una presentazione già firmata.

- Lo studente genera una presentazione selettiva contenente l'attributo `degree`, firmata e corredata di proof;
- Un attaccante intercetta questa presentazione e ne modifica localmente il valore, ad esempio cambiando `"Computer Engineering Master's Degree"` in `"PhD in Snackology"`;
- Il verificatore riceve questa versione alterata della presentazione e tenta di verificarla, includendo il controllo di:
  - firma dell'attributore (Università emittente);
  - validità della proof Merkle;
  - coerenza con il nonce ricevuto;
- Il risultato atteso è che la verifica fallisca: la manipolazione rende la proof incoerente con la radice del Merkle Tree registrata sulla DLT.

Questo attacco dimostra l'importanza dell'integrità crittografica e dell'uso dei Merkle Tree per garantire che ogni singolo attributo non possa essere alterato senza invalidare l'intera prova.

```python
# --- 4. DATA TAMPERING ATTACK SIMULATION ---
print("\n\n--- 4. DATA TAMPERING ATTACK SIMULATION ---")
print("[TAMPERING ATTACK] An attacker modifies the 'degree' attribute...")

nonce_2 = os.urandom(16).hex()
tampered_presentation = student.generate_presentation(issuer_university.name, "degree", nonce=nonce_2)

if tampered_presentation:
    tampered_presentation["selected_attribute"]["degree"] = "PhD in Snackology"
    print(f"  - Attribute tampered to: '{tampered_presentation['selected_attribute']['degree']}'")
    tampering_result = verifier_university.verify_presentation(tampered_presentation, dlt, expected_nonce=nonce_2)
    print(f"\n---> Tampering attack result: {'VALID' if tampering_result else 'INVALID'}")
```

Listing 4.10: Simulazione di manipolazione dei dati.

```
--- 4. DATA TAMPERING ATTACK SIMULATION ---
[TAMPERING ATTACK] An attacker modifies the 'degree' attribute...

[PRESENTATION] Luigi Veniero is creating a selective presentation...
[PRESENTATION] Created presentation for 'degree' with nonce '05df8009...'.
  - Attribute tampered to: 'PhD in Snackology'

[VERIFICATION] University of Rennes is verifying a presentation...
  - STEP 0/4: Nonce '05df8009...' valid and never used before.
  - STEP 1/4: Digital Signature AUTHENTIC.
  - VERIFICATION ERROR: Merkle Proof not valid. Data potentially tampered.

---> Tampering attack result: INVALID
```

Figura 4.4: Output manipolazione dati: inserimento di un degree fasullo

Flusso di revoca della credenziale

Questa fase dimostra la revoca di una credenziale e il tentativo, da parte dello studente, di riutilizzarla dopo la revoca.

- **Revoca sulla DLT.** L'università emittente invoca `revoke_credential()` passando la radice Merkle della credenziale. Tale radice viene contrassegnata come *revocata* all'interno della DLT, rendendo pubblica l'invalidità della credenziale;
- **Tentativo di riuso della credenziale revocata.** Lo studente genera una nuova presentazione selettiva (`nonce_3`) basata sulla stessa credenziale ormai revocata;
- **Verifica da parte dell'università ospitante.** Il verificatore controlla la presentazione:

  - convalida firma, Merkle proof e nonce;
  - consulta la DLT per verificare lo stato della radice Merkle;

  Poiché la credenziale risulta revocata, la verifica deve restituire `INVALID`;

- **Esito.** Il messaggio di output indica l'insuccesso della verifica, confermando che il sistema impedisce l'uso di credenziali revocate.

Questo passo evidenzia come la gestione della revoca, integrata nella DLT, garantisca la sicurezza contro l'uso improprio di credenziali non più valide.

```
# --- 5. REVOCATION FLOW ---
print("\n\n--- 5. REVOCATION FLOW ---")
issuer_university.revoke_credential(credential["merkle_root"], dlt)

print("\n[REVOKED] The student tries to use the revoked credential...")
nonce_3 = os.urandom(16).hex()
presentation_after_revocation = student.generate_presentation(issuer_university.
    name, attribute, nonce=nonce_3)
revocation_result = verifier_university.verify_presentation(
    presentation_after_revocation, dlt, expected_nonce=nonce_3)
print(f"\n---> Verification result after revocation: {'VALID' if revocation_result
    else 'INVALID'}")
```

Listing 4.11: Utilizzo in condizioni di revoca delle credenziali

```
--- 5. REVOCATION FLOW ---

[REVOKE] University of Salerno is revoking credential with MTR: e53b5537c4...
[REVOKE] Credential revoked successfully on DLT.

[REVOKED] The student tries to use the revoked credential...

[PRESENTATION] Luigi Veniero is creating a selective presentation...
[PRESENTATION] Created presentation for 'degree' with nonce '9e5047f7...'.

[VERIFICATION] University of Rennes is verifying a presentation...
  - STEP 0/4: Nonce '9e5047f7...' valid and never used before.
  - STEP 1/4: Digital Signature AUTHENTIC.
  - STEP 2/4: Merkle Proof valid. Data INTACT.
  - VERIFICATION ERROR: Status of credential on DLT not valid: REVOKED.

---> Verification result after revocation: INVALID
```

Figura 4.5: Output revoca delle credenziali.

4.3 Analisi delle prestazioni

Per valutare l'efficienza del protocollo proposto, sono state misurate le prestazioni in due aree chiave: la dimensione dei dati scambiati e la latenza delle operazioni

crittografiche. I test sono stati eseguiti in un ambiente di simulazione locale su una macchina standard.

È fondamentale sottolineare che i risultati ottenuti, sebbene incoraggianti, rappresentano uno scenario ideale. Un'implementazione in un ambiente di produzione reale introdurrebbe ulteriori fattori che influenzerebbero significativamente le prestazioni complessive.

4.3.1 Risultati della simulazione

Le misurazioni sono state effettuate durante il flusso operativo standard: emissione di una credenziale completa, generazione di una presentazione selettiva per un singolo attributo e successiva verifica.

1. **Dimensione dei dati**. La dimensione dei dati crittografici generati è un fattore cruciale per l'efficienza della trasmissione e dell'archiviazione.

    - **Firma digitale (RSA 2048bit)**: la firma generata ha una dimensione fissa di 256byte. Questo rappresenta il principale overhead per ogni credenziale emessa;
    - **Merkle Proof**: la dimensione della prova di appartenenza è logaritmica rispetto al numero totale di attributi. Per una credenziale con 7 attributi, la prova per un singolo attributo ha una dimensione di 295 byte. Questo dimostra l'efficienza del sistema: la dimensione della prova rimane contenuta anche per credenziali complesse.

2. **Latenza delle operazioni**. La latenza è stata misurata per le tre fasi principali del protocollo:

    - **Tempo di emissione**: l'intero processo di creazione del Merkle Tree e della firma digitale della radice richiede in media **2.24ms**. Questa è un'operazione *una tantum* eseguita dall'università emittente;
    - **Tempo di generazione della presentazione**: la creazione di una presentazione selettiva da parte del wallet dello studente è estremamente rapida, richiedendo in media **0.40ms**;
    - **Tempo di verifica**: la verifica completa (*controllo del nonce*, *firma digitale*, *Merkle Proof* e dello *stato sulla DLT simulata*) è l'operazione più complessa e richiede in media **1.20ms**.

4.3.2 Analisi critica e differenze con ambiente reale

I risultati della simulazione sono volutamente ottimistici, poiché si concentrano sulla validazione del protocollo crittografico in un ambiente controllato. È fondamentale comprendere che questi valori non tengono conto delle complessità e delle latenze intrinseche di un sistema distribuito reale.

Latenza della DLT e finalità delle transazioni

Questa è la differenza più significativa. Nella nostra simulazione, le interazioni con la DLT sono chiamate a metodo istantanee. In una blockchain reale, ogni operazione di scrittura (come la registrazione di una MTR o una revoca) è una transazione che deve superare un processo complesso prima di essere considerata definitiva:

1. **Propagazione sulla Rete**: la transazione deve essere inviata a tutti i nodi della rete;
2. **Meccanismo di Consenso**: deve essere inclusa in un nuovo blocco. Il tempo per creare un blocco dipende dal meccanismo di consenso (es. i 10 minuti del *Proof-of-Work* di Bitcoin o i 12 secondi per slot del *Proof-of-Stake* di Ethereum);
3. **Finalità**: per proteggersi da riorganizzazioni della catena (forks), una singola conferma non è sufficiente. Nella pratica, si attende la conferma di multipli blocchi successivi (es. la convenzione dei 6 blocchi su Bitcoin) prima di considerare una transazione immutabile. Questo porta il tempo di attesa per una conferma sicura da decine di minuti a oltre un'ora.

Latenza di rete e overhead di protocollo

La nostra simulazione esegue tutte le operazioni localmente. In un sistema reale, ogni interazione tra lo studente e l'università (es. la richiesta con il *nonce* e l'invio della presentazione) avverrebbe su una rete, probabilmente tra nazioni diverse (Italia e Francia). Questo introduce:

- **Latenza Fisica (RTT)**: il tempo materiale che i dati impiegano per viaggiare attraverso la rete.
- **Overhead dei Protocolli Sicuri**: prima di scambiare qualsiasi dato, le due parti devono stabilire un canale sicuro tramite un protocollo come TLS. Questo richiede un *handshake* iniziale che comporta diversi scambi di messaggi, aggiungendo ulteriore latenza prima ancora che la presentazione venga inviata.

Throughput e scalabilità (costi di transazione)

Una DLT reale ha un limite fisico al numero di transazioni per secondo (TPS) che può processare (es. Bitcoin $\sim$7 TPS, Ethereum $\sim$15–30 TPS). La nostra simulazione non ha questo limite. In uno scenario con migliaia di studenti, questo collo di bottiglia avrebbe conseguenze dirette:

- **Congestione**: le transazioni verrebbero messe in coda, aumentando i tempi di attesa per l'emissione e la revoca delle credenziali.

- **Costi di Transazione (Gas Fees)**: su blockchain come Ethereum, la congestione crea un mercato delle commissioni. Per far processare la propria transazione più velocemente, un utente (o un'università) dovrebbe pagare una commissione (*gas*) più alta. Questo introduce un costo monetario operativo per ogni credenziale emessa o revocata, un fattore completamente assente nella simulazione.

Implicazioni su storage e Costi operativi (Gas)

Oltre alla latenza, i costi di storage (archiviazione on-chain) e di gas (costo computazionale delle operazioni) sono fattori critici per la sostenibilità di un sistema DLT. L'architettura del progetto è stata esplicitamente progettata per minimizzare questi costi;

- **Analisi dello storage**: il principio guida è la minimizzazione dei dati on-chain, come descritto nel WP2;
  - **Dati On-Chain (sulla DLT)**: vengono archiviati solo i dati strettamente necessari per la fiducia e la verificabilità: le chiavi pubbliche delle università, le Merkle Roots (hash di 32 byte) e gli stati di revoca;
  - **Dati Off-Chain**: tutti i dati personali e accademici dello studente rimangono nel suo wallet privato e non vengono mai scritti sulla DLT;
  - **Impatto**: questa scelta è cruciale. Evita di appesantire la blockchain con dati pesanti e sensibili, riducendo drasticamente i costi di storage e migliorando la privacy;
- **Analidi dei Costi (Gas)**: su blockchain come Ethereum, le operazioni di scrittura hanno un costo in gas, mentre quelle di lettura sono gratuite. Il nostro protocollo sfrutta questo modello in modo efficiente;
  - **Operazioni di scrittura (a pagamento)**: l'emissione (registrazione della MTR) e la revoca sono le uniche operazioni che richiedono una transazione e quindi un costo in gas. Sono operazioni a basso costo (scrittura di un hash) e, nel caso della revoca, poco frequenti;
  - **Operazioni di lettura (gratuite)**: la verifica, che è l'operazione più frequente, richiede solo di leggere dati (la chiave pubblica e lo stato della MTR) dalla DLT. Questa operazione è gratuita e non consuma gas;

Questa ottimizzazione rende il sistema economicamente sostenibile, concentrando i costi sulle azioni meno comuni e rendendo gratuita l'operazione di verifica quotidiana.

Conclusioni simulazione

Quindi, mentre la simulazione dimostra che le operazioni crittografiche del protocollo sono efficienti, le prestazioni di un'implementazione reale sarebbero dominate dai tempi e dai limiti imposti dalla tecnologia DLT e dalla rete sottostante.

```python
# --- 6. PERFORMANCE ANALYSIS ---
print("\n\n" + "="*30)
print("       PERFORMANCE ANALYSIS")
print("="*30)

signature_size = len(credential['signature'])
proof_size = len(json.dumps(presentation['proof_of_membership']))

print("\n1. Data Size (approx. in bytes):")
print(f"   - RSA Signature (2048-bit): {signature_size} bytes")
print(f"   - Merkle Proof (1 attribute out of {len(student_data)}): {proof_size}
    bytes")

print("\n2. Latency of Cryptographic Operations:")
print(f"   - Credential Issuance (Merkle Tree + Signature): {issuance_time:.2f} ms"
    )
print(f"   - Selective Presentation Generation: {presentation_time:.2f} ms")
print(f"   - Full Verification (Nonce + Signature + Proof + DLT): {
    verification_time:.2f} ms")
```

Listing 4.12: Implementazione del codice usato per l'analisi delle prestazioni.

```
==============================
       PERFORMANCE ANALYSIS
==============================

1. Data Size (approx. in bytes):
   - RSA Signature (2048-bit): 256 bytes
   - Merkle Proof (1 attribute out of 7): 295 bytes

2. Latency of Cryptographic Operations:
   - Credential Issuance (Merkle Tree + Signature): 2.54 ms
   - Selective Presentation Generation: 0.56 ms
   - Full Verification (Nonce + Signature + Proof + DLT): 0.94 ms
```

Figura 4.6: Output log analisi delle performance

Capitolo 5

WPEnd: Conclusioni

Responsabile WP End:	Michele Penna, Gabriele Imparato

Il progetto ha presentato un sistema per la gestione sicura e decentralizzata delle credenziali accademiche digitali, rivolto in particolare agli studenti Erasmus. L'architettura proposta combina tecnologie avanzate come i Merkle Tree, la crittografia asimmetrica e DLT per garantire integrità, autenticità, privacy, non ripudio e revocabilità delle credenziali.

- **Risultati chiave**:
  - **Sicurezza e Privacy**:
    - **Autenticità**: le firme digitali RSA assicurano che le credenziali provengano esclusivamente dalluniversità emittente;
    - **Non ripudio**: l'uso di firme crittografiche impedisce allemittente di negare lemissione delle credenziali;
    - **Privacy**: la divulgazione selettiva tramite Merkle Tree permette agli studenti di condividere solo gli attributi necessari, rispettando il principio di minimizzazione dei dati;
    - **Integrità**: la struttura del Merkle Tree e l'hashing crittografico (SHA-256) garantiscono che i dati non siano alterabili senza invalidare la credenziale;
    - **Revocabilità**: il sistema permette di invalidare tempestivamente le credenziali compromesse o non più valide tramite registrazione immutabile sulla DLT, garantendo che non possano essere riutilizzate fraudolentemente;
  - **Decentralizzazione**: la DLT elimina la necessità di un'autorità centrale, riducendo i rischi di censura e aumentando la trasparenza del sistema;
  - **Efficienza**: le simulazioni hanno dimostrato che le operazioni crittografiche sono rapide e scalabili, sebbene in un ambiente reale la latenza della DLT e i costi di transazione possano rappresentare sfide aggiuntive;

- **Punti di forza**:
  - **Resistenza agli attacchi**: il sistema è progettato per mitigare minacce come replay attack, man-in-the-middle e falsificazione delle credenziali;
  - **Controllo dello studente**: gli studenti hanno pieno controllo sulle proprie credenziali, in linea con i principi di self-sovereign identity;
  - **Interoperabilità**: l'adozione di standard aperti facilita l'integrazione con altre piattaforme e istituzioni;
- **Limiti e sviluppi futuri**:
  - **Prestazioni in ambiente reale**: le latenze della DLT e i costi di transazione potrebbero richiedere ottimizzazioni, come l'adozione di protocolli di consenso più efficienti;
  - **Usabilità**: l'esperienza utente, specialmente nella gestione del wallet fisico, potrebbe essere migliorata con interfacce più intuitive;
  - **Scalabilità**: saranno necessari test su larga scala per valutare la performance del sistema con un numero elevato di utenti.

In conclusione, il progetto rappresenta una simulazione di sistema realmente esistente sicuro, trasparente e rispettoso della privacy nell'ambito dello scambio di studenti tra università grazie al programma Erasmus+. Le soluzioni proposte offrono un equilibrio tra sicurezza e praticità, ponendo le basi per future implementazioni in contesti reali. Ulteriori ricerche potrebbero esplorare l'integrazione con altre tecnologie emergenti, come gli zero-knowledge proof, per migliorare ulteriormente la privacy e l'efficienza.

Bibliografia

[1] bit2me. *Cosa sono gli Hardware Wallet?* URL: https://academy.bit2me.com/it/portafogli-hardware/.

[2] Investopedia. *Merkle Tree in Blockchain: What It Is and How It Works*. URL: https://www.investopedia.com/terms/m/merkle-tree.asp.

[3] Wikipedia. *Hyperledger*. URL: https://it.wikipedia.org/wiki/Hyperledger.